

Neuro-Controller Design For Nonlinear Fighter Aircraft Maneuver Using Fully Tuned RBF Networks

Y. Li N. Sundararajan¹ P. Saratchandran

School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798

Abstract

In this paper, an on-line learning neuro-control scheme that incorporates a Growing Radial Basis Function Network (GRBFN) is proposed for a nonlinear aircraft controller design. The scheme is based on feedback-error-learning strategy in which the Neuro-Flight-Controller (NFC) augments a conventional controller in the loop. By using the Lyapunov synthesis approach, the tuning rule for updating all the parameters of the RBFN (weights, widths and centers of the Gaussian functions) is derived which ensures the stability of the overall system with improved tracking accuracy. The theoretical results are validated using simulation studies based on a nonlinear 6-DOF high performance fighter aircraft undergoing a high α stability-axis roll maneuver. Compared with a traditional RBFN where only the weights are tuned, a GRBFN with tuning of all the parameters can implement a more compact network structure with smaller tracking error.

Key words: Adaptive control, Feedback-error-learning, Lyapunov stability theory, Neuro-flight-controller (NFC), Radial basis function network (RBFN).

1 Introduction

With the stringent requirements and complexities of present day high performance fighter aircraft control systems, it is difficult to model or estimate the system nonlinearities accurately. For example, in performing the Herbst maneuver (Herbst,1990), the nonlinearities come as the consequence of the (nonlinear) aerodynamics, nonlinear inertial and kinematic couplings due to

¹ Corresponding author. E-mail: ensundara@ntu.edu.sg.

the high angle of attack and high angular roll rates. Although a conventional fixed gain or gain scheduled controller may assure stability for this maneuver, the tracking performance will be poor under severe unknown nonlinearities. Hence, new design tools have to be explored to control these systems.

Due to their powerful ability of approximating nonlinear functions, control laws and design methods incorporating Artificial Neural Networks (ANNs) have been extensively studied in the area of nonlinear adaptive control (Miller, Sutton, & Werbos, 1990)(Ge, Lee, & Harris, 1998)(Lewis, Jagannathan, & Yesilidrek, 1999). Among the various neural network paradigms, Radial Basis Function Networks (RBFN) with Gaussian function have been demonstrated to be more capable in nonlinear control since the early nineties. The popularity of RBFN is due to its good global generalization ability and a simple network structure that avoids unnecessary and lengthy calculations (Kadiramanathan, & Niranjana, 1993). For on-line implementation, Gomi and Kawato (1993) proposed a “feedback-error-learning” strategy, where a Gaussian RBFN is used for learning the inverse dynamics of a robotic system.

In (Calise, & Rysdyk, 1998), Calise and Rysdyk have summarized use of neural networks for flight control system design, with emphasis on nonlinear adaptive control. It has been shown that neural networks with on-line learning capabilities can adapt to changes in aircraft dynamics and hence provide a good performance for aircraft undergoing highly nonlinear maneuvers. However, in aircraft control, there are only limited papers that explore the application of RBF network. Singh and Wells (1995) used a RBFN to suppress wing rock for a slender delta wing configuration. Kim and Calise presented the use of RBFN to capture variations in Mach number in (Kim, & Calise, 1997), as these variations are difficult to be represented by polynomial functions in the transonic region. In all the above-mentioned applications, a classical approach to the RBFN controller’s implementation is utilized, which is to fix the number of hidden neurons, centers and widths of the Gaussian functions first and then estimate its weights according to the feedback error. With the option of only tuning the weights, it is difficult to select the centers and widths for the Gaussian functions without prior training and the inaccuracy caused by the misplaced centers and widths will result in performance degradation. To overcome these problems, in this paper, an on-line control strategy that originated from Kawato’s feedback-error-learning is proposed, in which all the parameters of the network (including the centers and widths) will be updated, so that the changes in the system dynamics can be captured quickly. Using the Lyapunov synthesis approach, an updating rule for this fully tuned RBFN is derived that ensures the stability of the overall system. In addition, the robustness of the proposed tuning law is also analyzed.

The performance of the proposed control scheme along with the tuning law is evaluated for a flight controller design based on a high performance fighter

aircraft executing a stability-axis roll maneuver (also referred to as a velocity vector roll) at high angle-of-attack. The stability-axis roll maneuver consists of a vertical pull-up to increase and maintain a large angle of attack, followed by a rapid stability axis roll of 180° and is completed by restoring the angle of attack to the original trim condition. This maneuver is similar to the Herbst maneuver, although to a lesser extent due to the loss of control moments in deep stall conditions and the absence of thrust vector control in the aircraft used in this study. It is difficult to control such a maneuver because the normal acceleration tends to draw the heavy nose and tail portions of the aircraft farther from the axis of rotation at high roll rates, resulting in a positive pitch rate and an increasing α . The importance in designing a neural controller to control this maneuver lies in the fact that it does exercise the aircraft to a very wide dynamic range in a short time and brings out all the nonlinearities of the aircraft.

For validating the proposed neural controller and to assess the performance of the aircraft under large amplitude maneuvers, a full-fledged 6-DOF nonlinear aircraft flight simulation package has been developed on a PC using SIMULINK under the MATLAB environment (Zhang, Wang, & Sundararajan, 1997). For high fidelity, original wind tunnel test data is used and the aircraft dynamics is simulated in its original nonlinear form(Nguyen, *et al.*, 1979). The performance capabilities of the designed controller for executing the high α stability-axis roll maneuver is presented based on the results from this simulation package. The results indicate that compared with a traditional RBFN where only the weights are tuned, the proposed neuro-controller where all the parameters are updated can implement a more compact network with smaller tracking error.

The rest of the paper is organized as follows: Section 2 describes the fully tuned RBFN control architecture together with the derivation of the stable adaptive tuning rule. Section 3 presents the detailed simulation results for this controller based on a high performance fighter aircraft model executing a large α velocity vector roll maneuver. Section 4 summarizes the conclusions from this study.

2 Derivation of Stable On-Line Tuning Rule for a fully tuned RBF network

2.1 Problem Formulation

The aircraft dynamics is represented by a continuous system,

$$\Sigma : \quad \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad (1)$$

In Eq.(1), smoothness of $\mathbf{f}()$ is assumed. \mathbf{x} is a bounded $n \times 1$ state vector, \mathbf{u} is a bounded $p \times 1$ control vector. Without loss of generality, it is assumed that $\mathbf{f}(\mathbf{0}, \mathbf{0}) = \mathbf{0}$, so that the origin is an equilibrium state. In general, the number of control inputs is less than that of the states ($p < n$). It is known that under this condition only the p states can be tracked perfectly. Partitioning \mathbf{x} into \mathbf{x}_t and \mathbf{x}_r , where \mathbf{x}_t refers to the p states that can be tracked perfectly, and \mathbf{x}_r refers to the rest of the states, the aircraft dynamics can be written as,

$$\Sigma : \quad \begin{pmatrix} \dot{\mathbf{x}}_t \\ \dot{\mathbf{x}}_r \end{pmatrix} = \begin{pmatrix} \mathbf{f}_t(\mathbf{x}, \mathbf{u}) \\ \mathbf{f}_r(\mathbf{x}, \mathbf{u}) \end{pmatrix} \quad (2)$$

The problem studied here is to design a neuro-controller such that the given p states $\mathbf{x}_t \in \mathbf{x}$ of the aircraft can track the desired commands \mathbf{x}_{dt} accurately, while at the same time the other states $\mathbf{x}_r \in \mathbf{x}$ asymptotically approach a equilibrium point \mathbf{x}_{dr} at the end of the maneuver.

Determining the conditions under which a solution $\mathbf{u}_d(t)$ exists forms part of the theoretical control problem. Based on the earlier studies in (Narendra, & Mukhopadhyay, 1997), to accomplish the above control objective, the system should satisfy Assumption 1.

Assumption 1: $\mathbf{f}_t(\mathbf{x}, \mathbf{u})$ has continuous bounded partial derivative in a certain neighborhood of all the points along the desired trajectory \mathbf{x}_d and the matrix $\partial \mathbf{f}_t(\mathbf{x}, \mathbf{u}) / \partial \mathbf{u}^T$ is nonsingular.

It follows from the Implicit Function Theorem that the desired $\mathbf{u}_d(t)$ can be expressed as,

$$\mathbf{u}_d(t) = \bar{\mathbf{f}}_t(\mathbf{x}_d, \dot{\mathbf{x}}_d) \quad (3)$$

where $\bar{\mathbf{f}}_t$, a $p \times 1$ smooth function is the inverse function of \mathbf{f}_t , and $\mathbf{x}_d = [\mathbf{x}_{dt}^T, \mathbf{x}_{dr}^T]^T$.

2.2 Control Strategy

An on-line control strategy, which is similar to the well-known “feedback-error-learning” scheme, is proposed in Fig.1. This scheme is adopted because it has the advantage of generating the desired input signal without requiring that the network be trained initially off-line. In this case, the outputs of the neuro-controller \mathbf{u}_{nn} depend on the desired input profile.

\mathbf{x} has been divided into the p states \mathbf{x}_t and the $n - p$ states \mathbf{x}_r . Assuming that all the states of the system are accessible, the tracking error \mathbf{e} is defined as $\mathbf{e} = \mathbf{x} - \mathbf{x}_d$, where $\mathbf{x}_d = [\mathbf{x}_{dt}^T, \mathbf{x}_{dr}^T]^T$. This partition has been used for analyzing the condition for the existence of the solution. In the following sections, they are combined together to derive the parameter tuning rule.

The error dynamics for the overall system is,

$$\dot{\mathbf{e}} = \dot{\mathbf{x}} - \dot{\mathbf{x}}_d = \mathbf{f}(\mathbf{x}, \mathbf{u}) - \mathbf{f}(\mathbf{x}_d, \mathbf{u}_d) \quad (4)$$

Using Taylor series expansion and neglecting all the higher order terms,

$$\dot{\mathbf{e}} \simeq \mathbf{A}(t)\mathbf{e} + \mathbf{B}(t)(\mathbf{u} - \mathbf{u}_d) \quad (5)$$

where $\mathbf{A}(t) = \partial \mathbf{f}(\mathbf{x}, \mathbf{u}) / \partial \mathbf{x}^T|_{\mathbf{x}_d, \mathbf{u}_d}$, and $\mathbf{B}(t) = \partial \mathbf{f}(\mathbf{x}, \mathbf{u}) / \partial \mathbf{u}^T|_{\mathbf{x}_d, \mathbf{u}_d}$.

In this study, the control scheme consists of a neural controller and a conventional controller as shown in Fig.1. The conventional controller is used to make the closed loop system stable along the desired trajectory. This conventional controller can be realized by any approach like PID control, linear quadratic regulator (LQR), or H_∞ controller, etc. Here, a proportional controller satisfying Assumption 2 is used.

Assumption 2: The closed loop dynamic system, whose feedback controller is designed based on the nominal aircraft model, is stable along the desired flight trajectory.

With only a linear proportional controller $\mathbf{u} = \mathbf{K}_p(t)\mathbf{e}$, the error dynamics is,

$$\dot{\mathbf{e}} \simeq (\mathbf{A}(t) + \mathbf{B}(t)\mathbf{K}_p(t))\mathbf{e} - \mathbf{B}(t)\mathbf{u}_d \quad (6)$$

For satisfying Assumption 2, $\mathbf{K}_p(t)$ is properly designed so that $\mathbf{J}(t) = \mathbf{A}(t) + \mathbf{B}(t)\mathbf{K}_p(t)$ is stable. In Fig.1 for the neural controller a RBFN controller is used and the total control signal to the aircraft is the sum of the conventional controller and the RBFN controller signals as given by,

$$\mathbf{u} = \mathbf{u}_{nn} + \mathbf{K}_p(t)\mathbf{e} \quad (7)$$

2.3 RBFN Approximation and Error Dynamics

Setting the RBFN's inputs $\underline{\zeta}$ to $\underline{\zeta} = [\mathbf{x}_d^T, \dot{\mathbf{x}}_d^T]^T$, \mathbf{u}_d can be approximated by an RBFN through on-line learning as,

$$\begin{aligned}
\mathbf{u}_d &= \mathbf{u}_{nn}^* + \underline{\epsilon}_h \\
&= \sum_{k=1}^h \mathbf{w}_k^* \exp\left(-\frac{1}{\sigma_k^{*2}} \|\underline{\zeta} - \underline{\mu}_k^*\|^2\right) + \underline{\epsilon}_h = \mathbf{W}^{*T} \underline{\phi}(\underline{\mu}^*, \sigma^*, \underline{\zeta}) + \underline{\epsilon}_h
\end{aligned} \tag{8}$$

where \mathbf{W}^* is the $h \times p$ optimal weight matrix (h indicates the number of hidden neurons) and $\underline{\phi}$ is a $h \times 1$ vector of Gaussian functions, which are determined by the optimal centers $\underline{\mu}^*$ and widths σ^* , for simplicity $\underline{\phi}^*$ is used instead of $\underline{\phi}(\underline{\mu}^*, \sigma^*, \underline{\zeta})$ hereafter. Using approximation theory, the inherent approximation error $\underline{\epsilon}_h$ can be reduced arbitrarily by increasing the number of hidden neurons h (Park, & Sandberg, 1991). Hence, it is reasonable to assume that $\underline{\epsilon}_h$ is bounded by a constant ϵ_H , and

$$\epsilon_H = \sup_{t \in R^+} \|\underline{\epsilon}_h(t)\| \tag{9}$$

If the variables \mathbf{x}_d and $\dot{\mathbf{x}}_d$ are in compact sets, ϵ_H is finite. If they are not in compact sets, a scaling mechanism can be utilized to convert them into a compact set (Fabri, & Kadiramanathan, 1996).

With the RBFN controller, from Eq.(7), the control input vector \mathbf{u} is,

$$\mathbf{u} = \sum_{k=1}^h \hat{\mathbf{w}}_k \exp\left(-\frac{1}{\hat{\sigma}_k^2} \|\underline{\zeta} - \hat{\underline{\mu}}_k\|^2\right) + \mathbf{K}_p(\mathbf{t})\mathbf{e} = \hat{\mathbf{W}}^T \hat{\underline{\phi}} + \mathbf{K}_p(\mathbf{t})\mathbf{e} \tag{10}$$

where $\hat{\mathbf{w}}_k$ is the estimated weights, $\hat{\sigma}_k$ and $\hat{\underline{\mu}}_k$ are the estimated center and width for the k th hidden neuron. $\hat{\mathbf{W}}$ and $\hat{\underline{\phi}}$ are the corresponding matrix and vector expressions. Substituting Eq.(8), Eq.(10) into Eq.(5), defining $\mathbf{J}(\mathbf{t}) = \mathbf{A}(\mathbf{t}) + \mathbf{B}(\mathbf{t})\mathbf{K}_p(\mathbf{t})$ and $\hat{\tilde{\mathbf{W}}}$ and $\hat{\tilde{\underline{\phi}}}$ as:

$$\hat{\tilde{\mathbf{w}}}_i = \hat{\mathbf{w}}_i^* - \hat{\mathbf{w}}_i,$$

$$\hat{\tilde{\underline{\phi}}} = \hat{\underline{\phi}}^* - \hat{\underline{\phi}}$$

and neglecting the higher order error terms of the form $\hat{\tilde{\mathbf{W}}}^T \hat{\tilde{\underline{\phi}}}$, the error dynamics is written as,

$$\dot{\mathbf{e}} \simeq \mathbf{J}(\mathbf{t})\mathbf{e} - \mathbf{B}(\mathbf{t})(\hat{\tilde{\mathbf{W}}}^T \hat{\tilde{\underline{\phi}}} + \tilde{\mathbf{W}}^T \hat{\underline{\phi}}) - \mathbf{B}(\mathbf{t})\underline{\epsilon}_h \tag{11}$$

where $\mathbf{B}(\mathbf{t})(\hat{\tilde{\mathbf{W}}}^T \hat{\tilde{\underline{\phi}}} + \tilde{\mathbf{W}}^T \hat{\underline{\phi}})$ represents the learning error E_l .

2.4 Stable Adaptive Tuning Rule for Fully Tuned RBFN

In this section, the updating rule for a fully tuned RBFN controller is derived based on the feedback-error-learning scheme.

Choose the following candidate Lyapunov function,

$$V = \frac{1}{2} \mathbf{e}^T \mathbf{P}(\mathbf{t}) \mathbf{e} + \frac{1}{2} \text{tr}(\tilde{\mathbf{W}}^T \Theta \tilde{\mathbf{W}}) + \frac{1}{2} \underline{\hat{\phi}}^T \Lambda \underline{\hat{\phi}} \quad (12)$$

where $\mathbf{P}(\mathbf{t})$ is an $n \times n$ time varying, symmetric, positive definite matrix, and Θ, Λ are $h \times h$ non-negative definite matrices. Using Eq.(11), the derivative of the Lyapunov function is given by,

$$\begin{aligned} \dot{V} \simeq & -\mathbf{e}^T \mathbf{Q}(\mathbf{t}) \mathbf{e} - \underline{\epsilon}_h^T \mathbf{B}(t)^T \mathbf{P}(\mathbf{t}) \mathbf{e} - \underline{\hat{\phi}}^T \hat{\mathbf{W}} \mathbf{B}(t)^T \mathbf{P}(\mathbf{t}) \mathbf{e} - \underline{\hat{\phi}}^T \tilde{\mathbf{W}} \mathbf{B}(t)^T \mathbf{P}(\mathbf{t}) \mathbf{e} \\ & + \text{tr}(\tilde{\mathbf{W}}^T \Theta \dot{\tilde{\mathbf{W}}}) + \underline{\hat{\phi}}^T \Lambda \dot{\underline{\hat{\phi}}} \end{aligned} \quad (13)$$

where $\mathbf{Q}(\mathbf{t}) = -\frac{1}{2}(\mathbf{J}(\mathbf{t})^T \mathbf{P}(\mathbf{t}) + \mathbf{P}(\mathbf{t}) \mathbf{J}(\mathbf{t}) + \dot{\mathbf{P}}(\mathbf{t}))$. Since $\mathbf{J}(\mathbf{t})$ is stable, the Lyapunov function can always be found and has a unique solution. Noting in Eq.(13),

$$\text{tr}(\tilde{\mathbf{W}}^T \Theta \dot{\tilde{\mathbf{W}}}) = \sum_{i=1}^p \tilde{\mathbf{w}}_i^T \Theta \dot{\tilde{\mathbf{w}}}_i \quad (14)$$

$$\underline{\hat{\phi}}^T \tilde{\mathbf{W}} \mathbf{B}(t)^T \mathbf{P}(\mathbf{t}) \mathbf{e} = \sum_{i=1}^p \underline{\hat{\phi}}^T \tilde{\mathbf{w}}_i \mathbf{B}_i^T \mathbf{P}(\mathbf{t}) \mathbf{e} \quad (15)$$

Eq.(13) becomes,

$$\begin{aligned} \dot{V} \simeq & -\mathbf{e}^T \mathbf{Q}(\mathbf{t}) \mathbf{e} - \underline{\epsilon}_h^T \mathbf{B}(t)^T \mathbf{P}(\mathbf{t}) \mathbf{e} + \underline{\hat{\phi}}^T (-\hat{\mathbf{W}} \mathbf{B}(t)^T \mathbf{P}(\mathbf{t}) \mathbf{e} + \Lambda \dot{\underline{\hat{\phi}}}) \\ & + \sum_{i=1}^p (-\tilde{\mathbf{w}}_i^T \underline{\hat{\phi}} \mathbf{B}_i^T \mathbf{P}(\mathbf{t}) \mathbf{e} + \tilde{\mathbf{w}}_i^T \Theta \dot{\tilde{\mathbf{w}}}_i) \end{aligned} \quad (16)$$

where $\dot{\tilde{\mathbf{w}}}_i$ is the i th column of matrix $\dot{\tilde{\mathbf{W}}}$, \mathbf{B}_i is the i th column of matrix $\mathbf{B}(t)$.

If $\dot{\tilde{\mathbf{w}}}_i$ and $\dot{\underline{\hat{\phi}}}$ are selected as,

$$\dot{\tilde{\mathbf{w}}}_i = \Theta^{-1} \underline{\hat{\phi}} \mathbf{B}_i^T \mathbf{P}(\mathbf{t}) \mathbf{e}, \quad i = 1, \dots, p \quad (17)$$

$$\dot{\underline{\hat{\phi}}} = \Lambda^{-1} \hat{\mathbf{W}} \mathbf{B}(t)^T \mathbf{P}(\mathbf{t}) \mathbf{e} \quad (18)$$

Eq.(16) becomes,

$$\dot{V} \simeq \mathbf{e}^T \mathbf{Q}(\mathbf{t}) \mathbf{e} - \underline{\epsilon}_h^T \mathbf{B}(t)^T \mathbf{P}(\mathbf{t}) \mathbf{e} \quad (19)$$

\dot{V} can be demonstrated negative using Eq.(9),

$$\dot{V} \leq -\|\mathbf{e}\| \lambda_{\min}(\mathbf{Q}(\mathbf{t})) \|\mathbf{e}\| + \|\mathbf{e}\| \lambda_{\max}(\mathbf{P}(\mathbf{t})) \|\mathbf{B}(t)\|_{\epsilon_H} \quad (20)$$

Let $\|\mathbf{B}(t)\|_{\epsilon_H} = \delta_{\epsilon_h}$, it can be derived directly that \dot{V} is negative when

$$\|\mathbf{e}\| \geq \frac{\lambda_{\max}(\mathbf{P}(\mathbf{t}))}{\lambda_{\min}(\mathbf{Q}(\mathbf{t}))} \delta_{\epsilon_h} = E_a \quad (21)$$

Using the universal approximation proposition (Park, & Sandberg, 1991), by increasing the number h , ϵ_H can be reduced arbitrarily small, which means that E_a tends to zero when $h \rightarrow \infty$ and the negativeness of the Lyapunov function can be guaranteed, resulting in the overall system to be stable. However, it should be noted that in the real implementation, $h \rightarrow \infty$ is impossible and in this case only the uniform ultimate boundedness (UUB) of the error signals can be achieved.

Since $\dot{\hat{\mathbf{w}}}_i = \dot{\mathbf{w}}_i^* - \dot{\hat{\mathbf{w}}}_i$, $\dot{\hat{\underline{\phi}}} = \dot{\underline{\phi}}^* - \dot{\hat{\underline{\phi}}}$ and $\dot{\mathbf{w}}_i^* = \mathbf{0}$, $\dot{\underline{\phi}}^* = \mathbf{0}$, the tuning rules are,

$$\dot{\hat{\mathbf{w}}}_i = -\Theta^{-1} \hat{\underline{\phi}} \mathbf{B}_i^T \mathbf{P}(\mathbf{t}) \mathbf{e}, \quad i = 1, \dots, p \quad (22)$$

$$\dot{\hat{\underline{\phi}}} = -\Lambda^{-1} \hat{\mathbf{W}} \mathbf{B}(t)^T \mathbf{P}(\mathbf{t}) \mathbf{e} \quad (23)$$

2.5 Robustness Analysis

From the above derivation it can be seen that using the proposed method, provided Assumption 2 stands, the nonlinear function $\mathbf{f}()$ need not be known exactly. On the other hand, since the tuning rule derived involve the prior knowledge of $\mathbf{B}(t)$, the robustness of the tuning law is analyzed in case $\mathbf{B}(t)$ is partially known or unknown (model error or actuator failure).

Case 1: $\mathbf{B}(t)$ is varying, but satisfies $\mathbf{B}(t) = k(t)\mathbf{B}_0$, where \mathbf{B}_0 is a known nominal value, $k(t)$ is a positive scalar varying with time.

Provided the weight tuning rule and the tuning rule of $\underline{\phi}$ are given separately as,

$$\dot{\hat{\mathbf{w}}}_i = -k(t)\mathbf{\Theta}^{-1}\hat{\underline{\phi}}\mathbf{B}_{0i}^T\mathbf{P}(\mathbf{t})\mathbf{e}, \quad i = 1, \dots, p \quad (24)$$

$$\dot{\underline{\hat{\phi}}} = -k(t)\mathbf{\Lambda}^{-1}\hat{\mathbf{W}}\mathbf{B}_0^T\mathbf{P}(\mathbf{t})\mathbf{e} \quad (25)$$

\dot{V} can be demonstrated negative using the same condition as in Eq.(21). In implementation, $k(t)$ can be replaced by a positive scalar η and the system's convergence characteristic will not change.

Case 2: $\mathbf{B}(t)$ is unknown and is represented by a nominal value plus its variation, $\mathbf{B}(t) = \mathbf{B}_0(\mathbf{I} + \mathbf{\Delta B}(t))$.

Assume $\|\mathbf{\Delta B}(t)\| \leq M$, M is the upper bound for the uncertainty. The derivative of Lyapunov function of Eq.(13) is re-analyzed and it can be derived directly that \dot{V} is negative when,

$$\|\mathbf{e}\| \geq \frac{\|\mathbf{P}\|\|\mathbf{B}_0\|}{\lambda_{\min}(\mathbf{Q})}\delta'_{\epsilon_h} + \frac{\|\mathbf{P}\|\|\mathbf{B}_0\|}{\lambda_{\min}(\mathbf{Q})}\zeta(\underline{\hat{\phi}}, \tilde{\mathbf{W}})\|\mathbf{\Delta B}(t)\| = E_a + E_{lm} \quad (26)$$

In the above equation, E_a is caused by the approximation inaccuracy ϵ_h , and E_{lm} is caused by the product of the learning inaccuracy $E_l(\|(\underline{\hat{\phi}}^T\hat{\mathbf{W}} + \underline{\hat{\phi}}^T\tilde{\mathbf{W}})\|)$ and modeling variation $E_m(\|\mathbf{\Delta B}(t)\|)$.

Remarks:

(1) If there is neither approximation inaccuracy (i.e. $\epsilon_h = 0$) nor model variation (i.e. $\mathbf{\Delta B}(t) = 0$), \dot{V} is negative semidefinite and hence the stability of the overall scheme is guaranteed.

(2) The approximation inaccuracy E_a is related to the number of hidden neurons used in the RBFN. Given enough hidden neurons, this inaccuracy will be very small.

(3) The second item E_{lm} is a combination of the learning inaccuracy and modeling variation. The product of ζ and $\mathbf{\Delta B}(t)$ indicates that even under large model variation $\mathbf{\Delta B}(t)$, if the RBFN learns the desired value \mathbf{W}^* and $\underline{\hat{\phi}}^*$ correctly, E_{lm} is still very small.

2.6 Implementation of the Tuning Rule

In Eq.(23), the Gaussian function $\hat{\underline{\phi}}$ is embedded with the centers' locations $\hat{\underline{\mu}}$ and widths $\hat{\sigma}$, i.e. $\hat{\underline{\phi}} = \underline{\phi}(\hat{\underline{\mu}}, \hat{\sigma}, \underline{\zeta})$. Combining all the adaptable parameters into a big vector, $\underline{\chi} = [\hat{\mathbf{w}}_1^T, \hat{\mu}_1^T, \hat{\sigma}_1, \dots, \hat{\mathbf{w}}_h^T, \hat{\mu}_h^T, \hat{\sigma}_h]^T$, a simple updating rule for $\underline{\chi}$ can be derived. Because the real implementation has to be carried out in a discrete-time framework, the tuning laws derived earlier in a continuous framework have been discretised as shown below.

First, Eq.(22) can be converted into,

$$\begin{aligned}
\dot{\hat{\mathbf{w}}}_i &= -\mathbf{\Theta}^{-1} \hat{\underline{\phi}} \mathbf{B}_i^T \mathbf{P}(\mathbf{t}) \mathbf{e}, \quad i = 1, \dots, p \\
\Rightarrow \dot{\hat{\mathbf{W}}}^T &= -\mathbf{B}(t)^T \mathbf{P}(\mathbf{t}) \mathbf{e} \hat{\underline{\phi}}^T \Rightarrow \dot{\hat{\mathbf{w}}}_j = -\mathbf{B}(t)^T \mathbf{P}(\mathbf{t}) \mathbf{e} \hat{\underline{\phi}}_j, \\
j &= 1, \dots, h, \quad \mathbf{\Theta} = \mathbf{I}
\end{aligned} \tag{27}$$

where $\hat{\mathbf{w}}_j$, a column vector, is the j th row of matrix $\hat{\mathbf{W}}$. Define $\hat{\mathbf{g}} = \hat{\mathbf{W}}^T \hat{\underline{\phi}}$ the output of the RBFN, $\hat{\underline{\phi}}$ is the derivative of $\hat{\mathbf{g}}()$ to the weights $\hat{\mathbf{W}}$, this equation can be converted into a discrete form,

$$\hat{\mathbf{w}}_i(n+1) = \hat{\mathbf{w}}_i(n) - \eta_1 \frac{\partial \hat{\mathbf{g}}}{\partial \hat{\mathbf{w}}_i^T} \mathbf{B}(n)^T \mathbf{P}(n) \mathbf{e}(n), \quad i = 1, \dots, h \tag{28}$$

Next, from Eq.(23),

$$\Delta \hat{\underline{\phi}}(n) = \hat{\underline{\phi}}(n+1) - \hat{\underline{\phi}}(n) = -\eta_2 \mathbf{\Lambda}^{-1} \hat{\mathbf{W}} \mathbf{B}(n)^T \mathbf{P}(n) \mathbf{e}(n) \tag{29}$$

Since Gaussian function $\hat{\underline{\phi}} = \phi(\hat{\underline{\mu}}, \hat{\theta}, \hat{\zeta})$ and the RBFN's output $\hat{\mathbf{g}} = \hat{\mathbf{W}}^T \hat{\underline{\phi}}$, the updating laws for centers and widths of the basis functions become,

$$\begin{aligned}
\hat{\underline{\mu}}_i(n+1) &= \hat{\underline{\mu}}_i(n) + \eta_3 \frac{\partial \hat{\underline{\phi}}(n)}{\partial \hat{\underline{\mu}}_i^T} \Delta \hat{\underline{\phi}}(n) \\
&= \hat{\underline{\mu}}_i(n) - \eta_2 \eta_3 \frac{\partial \hat{\mathbf{g}}}{\partial \hat{\underline{\mu}}_i^T} \mathbf{B}(n)^T \mathbf{P}(n) \mathbf{e}(n),
\end{aligned} \tag{30}$$

$$\begin{aligned}
\hat{\sigma}_i(n+1) &= \hat{\sigma}_i(n) + \eta_4 \frac{\partial \hat{\underline{\phi}}(n)}{\partial \hat{\sigma}_i^T} \Delta \hat{\underline{\phi}}(n) \\
&= \hat{\sigma}_i(n) - \eta_2 \eta_4 \frac{\partial \hat{\mathbf{g}}}{\partial \hat{\sigma}_i^T} \mathbf{B}(n)^T \mathbf{P}(n) \mathbf{e}(n),
\end{aligned} \tag{31}$$

where $i = 1, \dots, h$

In the above equations, η_1, η_2, η_3 and η_4 are positive scalars to be selected properly. Integrating Eq.(28), Eq.(30) and Eq.(31) by using the vector $\underline{\chi}$, the updating rule is,

$$\underline{\chi}(n+1) = \underline{\chi}(n) - \eta \alpha(n) \mathbf{B}(n)^T \mathbf{P}(n) \mathbf{e}(n) \tag{32}$$

where η is learning rate. $\alpha(n) = \nabla_{\underline{\chi}} \hat{\mathbf{g}}(\underline{\zeta}_n)$ is the gradient of the function $\hat{\mathbf{g}}()$ with respect to the parameter vector $\underline{\chi}$ evaluated at $\underline{\chi}(n-1)$, and

$$\alpha(n) = [\hat{\underline{\phi}}_1, \hat{\underline{\phi}}_1 \frac{2\hat{\mathbf{w}}_1}{\hat{\sigma}_1^2} (\underline{\zeta}_n - \hat{\underline{\mu}}_1)^T, \hat{\underline{\phi}}_1 \frac{2\hat{\mathbf{w}}_1}{\hat{\sigma}_1^3} \|\underline{\zeta}_n - \hat{\underline{\mu}}_1\|^2, \hat{\underline{\phi}}_h],$$

$$\dots \hat{\phi}_h \frac{2\hat{\mathbf{w}}_h}{\hat{\sigma}_h^2} (\underline{\zeta}_n - \hat{\underline{\mu}}_h)^T, \hat{\phi}_h \frac{2\hat{\mathbf{w}}_h}{\hat{\sigma}_h^3} \|\underline{\zeta}_n - \hat{\underline{\mu}}_h\|^2]^T \quad (33)$$

Since the number of hidden neurons of the RBFN is fixed in this scheme, E_a is quite large at the initial learning period. If $\|\mathbf{e}\| < E_a$, then it is possible that $\dot{V} > 0$, which implies that the parameters of the network may drift to infinity. A common way to avoid this problem and ensure the convergence of approximation error is to incorporate a dead zone in the tuning rules (Fabri, & Kadirkamanathan, 1996). Thus the tuning rule is given as,

$$\underline{\chi}_{n+1} = \begin{cases} \underline{\chi}_n - \eta \alpha_n \mathbf{B}(n)^T \mathbf{P}(n) \mathbf{e}_n & \|\mathbf{e}\| \geq e_0 \text{ and } \|\mathbf{w}\| \leq h^{\frac{1}{2}} N \\ \underline{\chi}_n & \text{otherwise} \end{cases} \quad (34)$$

N is a positive scalar and $h^{\frac{1}{2}} N$ is an upper bound on $\|\mathbf{W}\|$ (Euclidean norm of weight vector), e_0 is selected as the required accuracy on the state error \mathbf{e} .

2.7 Growing RBFN (GRBFN)

To implement a fully tuned RBFN as discussed in the previous sections, the number of the hidden neurons should be selected through a trial-and-error procedure. In case of approximating the inverse dynamics of a very complicated system, a lot of time is needed and superfluous hidden neurons will be added to the network. To overcome this problem, a GRBFN is selected in the strategy. In this paper, the minimal resource allocating network (MRAN) developed in (Lu, Sundararajan, & Saratchandran, 1998) is used: a hidden neuron is recruited when all the following three criteria are satisfied. (a) $\|\mathbf{e}(n)\| = \|\mathbf{x}(n) - \hat{\mathbf{x}}(n)\| > E_1(E_{max})$ (b) $d(n) = \|\underline{\zeta}_n - \underline{\mu}_{nr}\| > E_2$ (c) $e_{rmsn}(n) = \sqrt{\sum_{i=n-(N_w-1)}^n \frac{\|\mathbf{e}_i\|^2}{N_w \times p}} > E_3$, where E_1, E_2, E_3 are thresholds to be selected *a priori*, $\underline{\mu}_{nr}$ is the center of the hidden neuron which has the closest distance to the current network inputs $\underline{\zeta}_n$. Once the new hidden unit is recruited, the associated parameters are given as: $\underline{\mu}_{h+1} = \underline{\zeta}_n$, $\sigma_{h+1} = \kappa \|\underline{\zeta}_n - \underline{\mu}_{nr}\|$, $\mathbf{w}_{h+1} = \mathbf{e}(n)$. Due to the localization property of the $(h+1)$ th new hidden neuron, the error will bring it back into the tunable area and E_a decreases since $\epsilon_H(h+1) < \epsilon_H(h)$.

Table 1

Mathematical symbols used for describing the aircraft dynamics

u, v, w	airspeed in the X, Y, Z axes of the body frame respectively;
p, q, r	roll, pitch, yaw rates about the X, Y, Z axes respectively;
ϕ, θ, ψ	roll, pitch, yaw angles about the X, Y, Z axes respectively;
I_x, I_y, I_z	moments of inertia about the X, Y, Z axes respectively;
I_{xz}	cross moment of inertia;
$C_{X,t}, C_{Y,t}, C_{Z,t}$	total aerodynamic force coefficients in X, Y, Z axes respectively;
$C_{l,t}, C_{m,t}, C_{n,t}$	total aerodynamic moment coefficients in X, Y, Z axes respectively;
\bar{q}, \bar{c}	dynamic pressure and mean chord respectively;
S, b, H_e	area of the wing; wing span; engine angular momentum;
α, β	angle-of-attack; sideslip angle;
$\dot{\mu}$	stability-axis roll rate;
V_t	total air speed;
a_n, a_y	normal and sideway acceleration respectively;
$\delta_h, \delta_a, \delta_r$	deflection angle of elevator, aileron, rudder respectively;
$\delta_{sb}, \delta_{lef}$	deflection angle of speed brake and leading edge flap;

3 Neuro-Flight-Controller Performance for a Fighter Aircraft Under a High α Stability-Axis Roll Maneuver

In this section, the performance results of the neuro-controller developed in section 2 is presented based on a nonlinear high performance fighter aircraft (similar to a F-16) model with the same dynamics as in Eq.(1) and executing a high α stability-axis roll maneuver (Herbst-like maneuver). For high fidelity, the simulation is carried out on a full fledged 6-DOF nonlinear aircraft simulation package which was developed using the SIMULINK package under the MATLAB environment (Zhang, Wang, & Sundararajan, 1997). Before presenting the simulation results, a brief description of the nonlinear aircraft model along with the stability-axis roll maneuver is given.

3.1 Fighter Aircraft Model

The candidate aircraft is a high performance fighter aircraft similar to a F16 as reported in (Nguyen, Marilyn, William, Kemper, Philip, & Deal, 1979). The mathematical symbols used to describe the aircraft dynamics is given in Table 1.

Under the rigid body assumption and referenced to a body-fixed axis system, the mathematical equations describing the motions of the aircraft are,

Force equations:

$$\dot{u} = rv - qw - g\sin\theta + \frac{\bar{q}S}{m} C_{X,t} + \frac{T}{m} \quad (35)$$

$$\dot{v} = pw - ru + g\cos\theta\sin\phi + \frac{\bar{q}S}{m} C_{Y,t} \quad (36)$$

$$\dot{w} = qu - pv + g\cos\theta\cos\phi + \frac{\bar{q}S}{m} C_{Z,t} \quad (37)$$

Moment equations:

$$\dot{p} = \frac{I_x - I_z}{I_x} qr + \frac{I_{xz}}{I_x} (\dot{r} + pq) + \frac{\bar{q}Sb}{I_x} C_{l,t} + \frac{H_e q}{I_x} \quad (38)$$

$$\dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{I_{xz}}{I_y} (r^2 - p^2) + \frac{\bar{q}S\bar{c}}{I_y} C_{m,t} - \frac{H_e r}{I_y} \quad (39)$$

$$\dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{I_{xz}}{I_z} (\dot{p} - qr) + \frac{\bar{q}Sb}{I_z} C_{n,t} + \frac{H_e q}{I_z}. \quad (40)$$

Auxiliary equations:

$$\alpha = \tan^{-1} \frac{w}{u} \quad (41)$$

$$\beta = \sin^{-1} \frac{w}{V_t} \quad (42)$$

$$\dot{\mu} = p\cos\alpha + r\sin\alpha \quad (43)$$

These equations are highly nonlinear and coupled. The total aerodynamic coefficients $C_{X,t}$, $C_{Y,t}$, $C_{Z,t}$, $C_{l,t}$, $C_{m,t}$ and $C_{n,t}$ used in the simulation were derived from low-speed static and dynamic wind tunnel tests as reported in (Nguyen, *et al.*, 1979). Total coefficient equations were used to sum the various aerodynamic contributions to a particular force or moment. For example, the X axis moment coefficient:

$$\begin{aligned} C_{l,t} = & C_l(\alpha, \beta, \delta_h) + \Delta C_{l,lef} \left(1 - \frac{\delta_{lef}}{25}\right) + \left[\Delta C_{l,\delta_a} + \Delta C_{l,\delta_a,lef} \left(1 - \frac{\delta_{lef}}{25}\right)\right] \frac{\delta_a}{20} \\ & + \Delta C_{l,\delta_r} \frac{\delta_r}{30} + \frac{b}{2V_t} \left\{ [C_{l_r}(\alpha) + \Delta C_{l_r,lef}(\alpha) \left(1 - \frac{\delta_{lef}}{25}\right)] r \right. \\ & \left. + [C_{l_p}(\alpha) + \Delta C_{l_p,lef}(\alpha) \left(1 - \frac{\delta_{lef}}{25}\right)] p \right\} + \Delta C_{l_\beta}(\alpha) \beta. \end{aligned} \quad (44)$$

Other coefficients are given in similar forms (refer to (Nguyen, *et al.*, 1979)).

The aerodynamic moment coefficients are referenced to the nominal center of gravity location of $X_{cg} = 0.35\bar{c}$ and are corrected to the desired center-of-gravity position in the coefficient equations. The aircraft is powered by an afterburning turbofan jet engine. A typical model for the engine simulation is given by describing the thrust responses to throttle inputs. The thrust values are given in tabular form for idle, military and maximum thrust values. Moreover, the dynamics of the actuator that drives the control surfaces is modeled as simple first-order transfer function. For details refer to (Nguyen, *et al.*, 1979).

For the candidate fighter aircraft used in this study, there is no provision of thrust vectoring of control and hence the maneuver has to be restricted to within the capabilities of the conventional control surfaces. Since the leading edge flaps δ_{lef} are designed as auto-scheduled according to α , the available control surfaces are δ_h , δ_a and δ_r only. Also, in a high α stability-axis roll maneuver, throttle is usually pushed to nearly full position to prevent significant loss of speed, so the engine thrust control is not considered in the controller design. As can be seen from the aircraft dynamics, the mathematical equations describing the motions of the aircraft are the same as Eq.(1), where the state vector $\mathbf{x} = [u, v, w, p, q, r, \alpha, \beta, \dot{\mu}]^T$ with $\mathbf{x}_t = [\alpha, \beta, \dot{\mu}]^T$, and $\mathbf{u} = [\delta_h, \delta_a, \delta_r]^T$. The complete aerodynamic data acquired from the wind tunnel test covers a wide range of $-20^\circ < \alpha < 90^\circ$, $-30^\circ < \beta < 30^\circ$.

3.2 The High α Stability-axis Roll Maneuver

The high α stability-axis roll maneuver starts with a straight level flight at the trim condition of $\alpha = 2.37^\circ$, $V_t = 152.4m/s$, $h = 1000m$, $\delta_h = -0.5803^\circ$ (position A). A pitch-up command to the elevator is given to increase the α from its trim value to 30° at $3s$. The command for stability roll rate $\dot{\mu}$ starts at $5s$ and consists of three regions: the rising region (approximately $1s$), the holding region ($30^\circ/sec$ for approximately $3s$) and the arrest region (approximately $1s$). The stability axis roll rotates the aircraft about its stability axis so as to turn the direction of flight. Then α is decreased to bring the nose down to the initial α at $18s$. This maneuver is similar to the Herbst maneuver (Herbst, 1990), although to a lesser extent due to the loss of control moments in deep stall conditions and the absence of thrust vector control in the aircraft under study. The importance of designing an adaptive nonlinear controller to control the above maneuver lies in the fact that this maneuver does exercise the aircraft to a very wide dynamic range in a short time and it brings out all the nonlinearities of the aircraft.

3.3 Simulation Study Results

A full-fledged 6-DOF nonlinear aircraft flight simulation package (Zhang, *et al.*, 1997) has been developed using MATLAB/SIMULINK. For high fidelity, the wind tunnel test data is used and the aircraft dynamics is simulated in its original nonlinear form. The performance capabilities of the designed controller for executing the high α stability-axis roll maneuver is presented based on the results from this simulation package. These results are also used to check for any undesirable nonlinear effects like actuator saturation, etc. In this study, first a conventional proportional controller is designed based on a linearised model as a baseline control to achieve the desired response characteristics like rise time, overshoot, etc. Trimming and linearizing of the nonlinear model is carried out to obtain linear models suitable for conventional controller design. To satisfy the Assumption 2, a high α , low speed trim condition is selected as the design point for this conventional controller: $V_t = 70m/s, \alpha = 15.6474^\circ, h = 1000m, \delta_h = 0.5851^\circ$). Based on this linearised model, a proportional controller with a constant K_p is designed. Even though the aircraft dynamics is time varying, this constant proportional controller is able to maintain stability satisfying assumption 2. Since K_p is a constant matrix, P will also be a constant matrix in this case. Before proceeding further, it is instructive to evaluate the performance of the conventional controller alone. The conventional controller is designed to provide good control performance with the linearised aircraft model at the point A. Fig.2 presents the time histories of the angle of attack α , stability axis roll rate $\dot{\mu}$ and the side slip angle β . The first two variables have to track the commands closely and the sideslip angle should not deviate too much ($< 1^\circ$) from the initial trim value of 0° . The performance of the conventional controller when implemented in the 6-DOF nonlinear aircraft model is shown in Fig.2. Although this controller performs quite well for the linear model, it can be seen from the figure that the tracking performance deteriorates greatly when applied to the 6-DOF nonlinear model (dash-dot line) and the deviation in β is high (5°).

Hence, the RBFN controller is added in parallel to learn the inverse dynamics of the system and increase the tracking accuracy. The proposed feedback-error-learning strategy and the derived stable tuning rule are utilized to implement this Herbst-like maneuver. Fig.3 presents the results of using the proposed neuro-flight-controller. A comparison of using the (proposed) full tuning strategy (GRBF) with that of tuning only the weights of the RBFN is also presented. For the RBFN controller with only weights tuning, the number of hidden neurons are fixed as 15 and the centers of the hidden units are determined as 0.25 according to a grid method described in (Fabri, & Kadiramanathan, 1996). From Fig.3 it is obvious that the performance is poor for the case of only tuning the weights (dash-dot line). However, by tuning all the parameters of the GRBFN, good tracking performance is achieved for both

of the α and $\dot{\mu}$ with small variation in β (less than 1° as required) during the maneuver (line). It is also noted that apart from better accuracy, less hidden neurons are recruited using the GRBFN – 12 as compared to the 15 used by the normal RBFN. Fig.4 displays the control signals, the continuous line representing the total control signals and the dashed line representing the neuro-controller's outputs. It can be seen from the figure that the control inputs generated are well within the actuator limits without saturation ($\delta_h(max) = \pm 25^\circ, \delta_a(max) = \pm 20^\circ, \delta_r(max) = \pm 30^\circ$). Also it can be seen how the neural control outputs change when the commands are executed indicating the learning process. Fig.5 presents the trajectory of the maneuver in 3 dimensions using the neuro controller with the fully tuned RBF neural network. The numbers on the trajectory line indicate the time sequence. It is clear from the figures that the proposed neuro-flight controller does a good job for this complicated nonlinear aircraft maneuver.

4 Conclusions

This paper has presented a stable on-line control strategy using GRBFN to design a nonlinear flight controller for a full-fledged fighter aircraft executing high α stability-axis roll maneuver. Lyapunov stability theory is used to derive the tuning rule for updating all the parameters of the GRBFN. Simulation studies based on a high performance fighter aircraft model demonstrate that with the developed control strategy and the proposed tuning rule, a more compact network can be realized with better performance.

References

- [Calise, A.J. and Rysdyk, R.T. (1998)] Nonlinear adaptive flight control using neural networks. IEEE Control Systems Magazine 18(6), 14-25.
- [Fabri, S. and Kadiramanathan, V. (1996)] Dynamic structure neural networks for stable adaptive control of nonlinear systems. IEEE Trans. on Neural Networks 7(5), 1151-1167.
- [Ge, S.S., Lee, T.H. and Harris, C.J. (1998)] *Adaptive Neural Network Control of Robotic Manipulators*. World Scientific, Singapore.
- [Gomi, H. and Kawato, M. (1993)] Neural network control for a closed-loop system using feedback-error-learning. Neural Networks 6(7), 933-946.
- [Herbst, W.A. (1990)] Future fighter technologies. AIAA Journal of Aircraft 17, 561-566.

- [Kadirkamanathan, V and Niranjan, M. (1993)] A function estimation approach to sequential learning with neural networks. *Neural Computation* 5, 954-975.
- [Kim, B.S. and Calise, A.J. (1997)] Nonlinear flight control using neural networks. *AIAA Journal of Guidance, Control, and Dynamics* 20(1), 26-33.
- [Lewis, F.L., Yesilidrek, A., and Jagannathan, S. (1999)] *Neural network control of robot manipulators and nonlinear systems*. Taylor and Francis. Philadelphia, PA.
- [Lu, Y., Sundararajan, N., and Saratchandran, P. (1998)] Performance evolution of a sequential minimal RBF neural network learning algorithm. *IEEE Trans. on Neural Networks* 9(2), 308-318.
- [Miller, W.T., Sutton, R.S., and Werbos, P.J. (1990)] *Neural Networks for Control*. MIT Press. Cambridge, MA.
- [Narendra, K.S. and Mukhopadhyay, S. (1997)] Adaptive control using neural networks and approximate models. *IEEE Trans. on Neural Networks* 8(3), 475-485.
- [Park, J. and Sandberg, I.W. (1991)] Universal approximation using radial basis function networks. *Neural Computation* 3, 246-257.
- [Singh, S.N. and Wells, W.R. (1995)] Direct adaptive and neural control of wing-rock motion of slender delta wings. *AIAA Journal of Guidance, Control, and Dynamics* 18(1), 25-30.
- [Zhang, W., Wang, J., and Sundararajan, N. (1997)] Robust controller design for a high performance fighter aircraft under large alpha stability-axis roll maneuver. *Proceedings of the eighth MINDEF-NTU joint R&D seminar*. 23-31. Singapore.
- [Nguyen, L.T., Marilyn, E.O., William, P.G., et. al. (1979)] Simulator study of stall/post stall characteristics of a fighter airplane with relaxed longitudinal static stability. *NASA Technical Paper 1538*. USA.

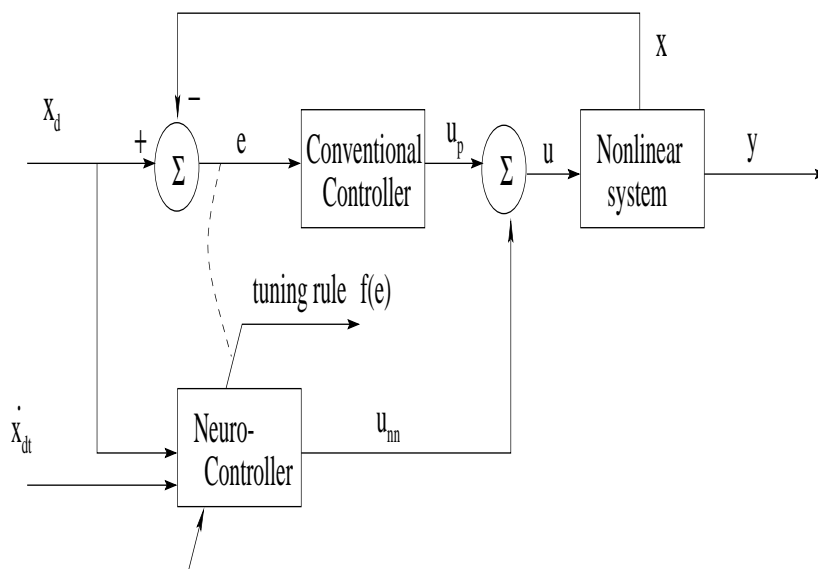


Fig. 1. Neural-flight-control (NFC) scheme

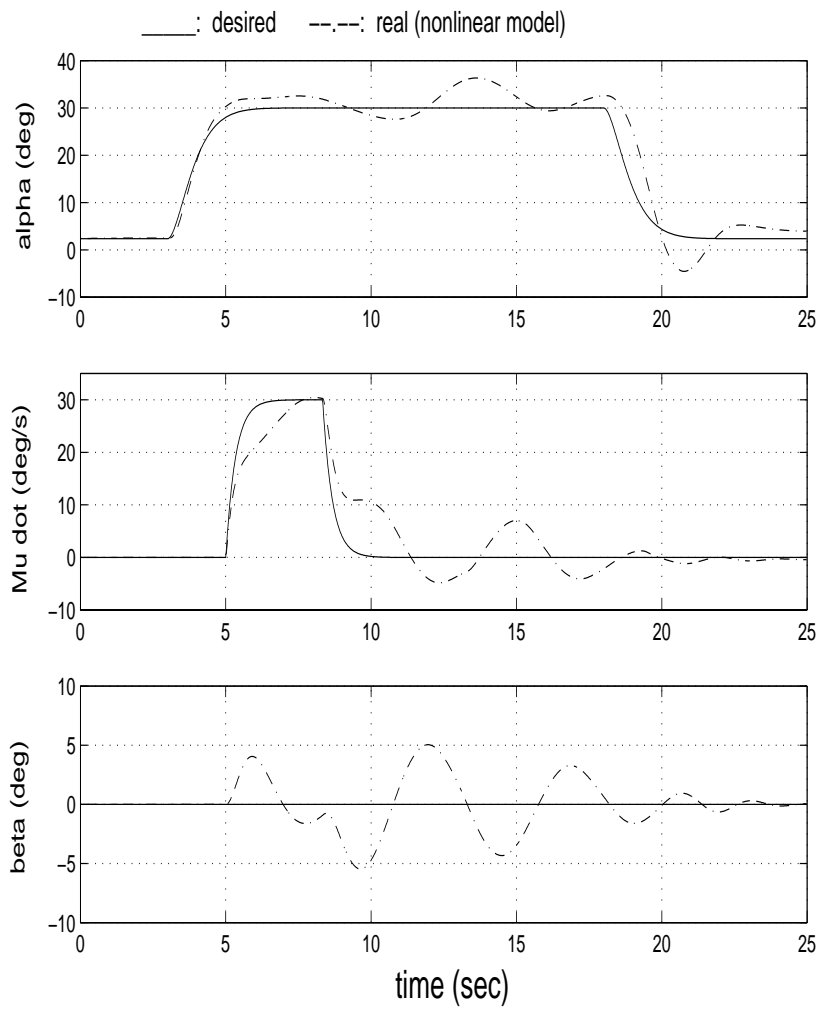


Fig. 2. Performance of the proportional controller for nonlinear aircraft model

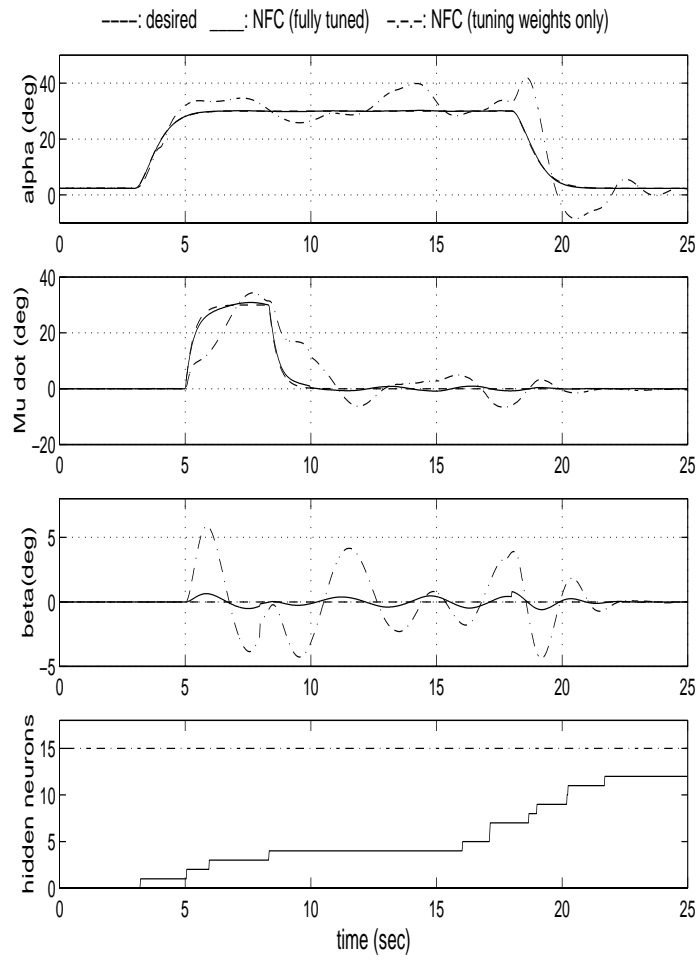


Fig. 3. Performance of the NFC for nonlinear aircraft model

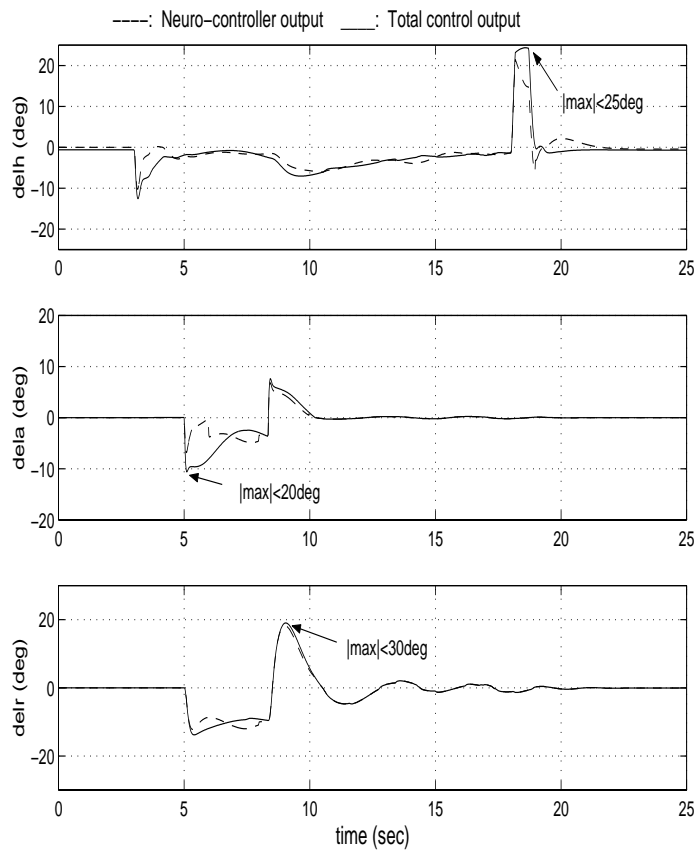


Fig. 4. Control Surface Deflections

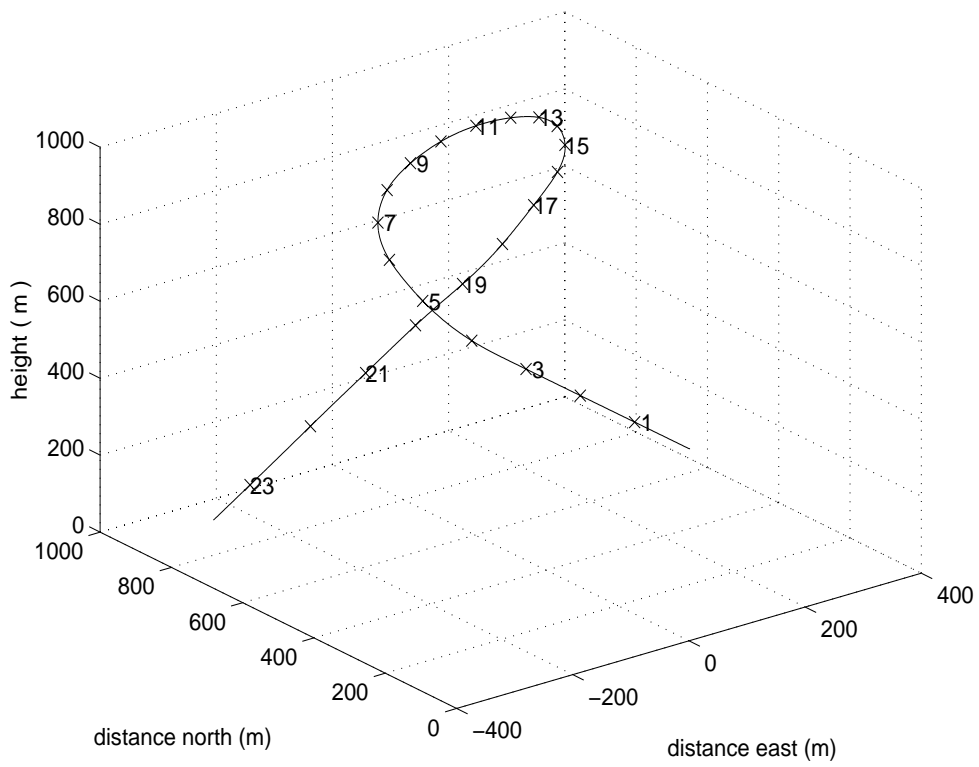


Fig. 5. Actual flight profile in three dimension (fully tuned GRBFN)